

Incorporating Tutorial Strategies into an Intelligent Assistant

Jim R. Davies, Abigail S. Gertner, Neal Lesh, Charles Rich, Jeff Rickel, Candace L. Sidner

TR2000-30 October 2000

Abstract

Computer tutors and intelligent software assistants have traditionally been thought of as different kinds of systems. However tutors and assistants share many properties. We have incorporated tutorial strategies into an intelligent assistant based on the COLLAGEN architecture. We are working on an agent, named Triton, which teaches and helps users with the graphical user interface of an air travel planning system. We found that the collaborative model underlying COLLAGEN is an excellent foundation for both an assistant and a tutor, and that both modes of interaction can be implemented in the same system with different parameter settings.

International Conference on Intelligent User Interfaces, Santa Fe, NM, January 2001

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Incorporating Tutorial Strategies into an Intelligent Assistant

Jim R. Davies
Abigail S. Gertner
Neal Lesh
Charles Rich
Jeff Rickel
Candace L. Sidner

TR-2000-30 October 2000

Abstract

Computer tutors and intelligent software assistants have traditionally been thought of as different kinds of systems. However tutors and assistants share many properties. We have incorporated tutorial strategies into an intelligent assistant based on the COLLAGEN architecture. We are working on an agent, named Triton, which teaches and helps users with the graphical user interface of an air travel planning system. We found that the collaborative model underlying COLLAGEN is an excellent foundation for both an assistant and a tutor, and that both modes of interaction can be implemented in the same system with different parameter settings.

Proceedings of Intelligent User Interfaces 2001, January 2001, Santa Fe, New Mexico USA

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Information Technology Center America; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Information Technology Center America. All rights reserved.

Submitted August 2000, revised and released October 2000.

Incorporating Tutorial Strategies Into an Intelligent Assistant

Jim R. Davies
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, jimmydavies@usa.net

Abigail S. Gertner
The MITRE Corporation
202 Burlington Rd, ms. K302
Bedford, MA 01730, gertner@mitre.org

Neal Lesh, Charles Rich,
Candace L. Sidner
Mitsubishi Electric Research Labs, 201 Broadway,
Cambridge, MA 02139, {lesh, rich, sidner}@merl.com

Jeff Rickel
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292-6695, rickel@isi.edu

ABSTRACT

Computer tutors and intelligent software assistants have traditionally been thought of as different kinds of systems. However tutors and assistants share many properties. We have incorporated tutorial strategies into an intelligent assistant based on the COLLAGEN architecture. We are working on an agent, named Triton, which teaches and helps users with the graphical user interface of an air travel planning system. We found that the collaborative model underlying COLLAGEN is an excellent foundation for both an assistant and a tutor, and that both modes of interaction can be implemented in the same system with different parameter settings.

Categories and Subject Descriptors

D.3.3 [Programming Languages]: Language Constructs and Features – *abstract data types, polymorphism, control structures*. This is just an example, please use the correct category and subject descriptors for your submission.

General Terms

Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification.

Keywords

Intelligent tutoring systems, software agents, intelligent assistants, collaboration, discourse

1. INTRODUCTION

As a student gains expertise, a good tutor's behavior often begins to resemble an assistant's. Similarly, a good assistant can teach tasks as well as perform them. We believe that, especially for procedural tasks, tutoring and assisting are best thought of as points on a spectrum of collaboration.

In this paper, we report on our work in progress to incorporate tutorial strategies into an intelligent assistant for an air travel planning application. Our initial findings are (1) that by viewing both tutoring and assisting as collaborative activities, the same set of mechanisms can be used to support both, and (2) that COLLAGEN can be extended and generalized to support both activities. This paper is a case study of how we extended COLLAGEN and used it to build a tutoring/assisting agent called Triton. We will describe some of the differences between tutoring and assisting, and discuss how these differences can be encoded as parameters, in a single system.

1.1 The COLLAGEN System

COLLAGEN (COLLaborative AGENT) [7] is middleware for building interface agents based on collaborative discourse theory. COLLAGEN provides application developers with a platform on which to implement agents for applications. The key algorithms and data structures in COLLAGEN come from recent work in collaborative discourse theory [5], which provides a general model of communication between user and agent (including interpreting intentions), and with a model of actions for manipulation of the application itself. The action model provides for action decomposition (via recipes) which the developer specifies for a particular application. The agent uses these recipes to interpret user actions and to plan its own actions.

As shown in Fig. 1, COLLAGEN provides user and agent "home" windows for communication purposes. Agents built in COLLAGEN interact with both the user and the application. Our agent, Triton, was built with COLLAGEN to assist or tutor the user in the use of an air travel application. Unlike COLLAGEN, typical authoring programs for Intelligent Tutoring Systems (ITS) require developers to intermix the encoding of domain knowledge, pedagogical strategies, and the application environment. These systems also do not model the communication and actions of the agent and user as a collaboration.

1.2 The Air Travel Planning Application

The air travel application can be used without an agent. Routes are selected by clicking on cities on a map. The time intervals during which the user needs to be in a particular city are specified with sliding bars.



Figure 1: A screenshot of the air travel application and agent. The agent's window is in the upper right corner. The user's window is in the lower left corner.

1.3 The Assistant

We have previously reported [7] on an intelligent assistant we built for the air travel application using COLLAGEN. The assistant monitors what the user does and communicates with her so that they have a shared understanding of the task. With this knowledge, the assistant can propose next actions or even perform them itself. Fig. 2 is an example scenario of the assistant's behavior (the city constraints of the trip were communicated earlier in the dialogue).

36. **User:** Let's plan the second leg of the trip.
37. **Assistant:** Okay.
38. **User:** [selects Washington] What next?
39. **Assistant:** [points to San Francisco] I will select San Francisco.
40. **User:** Okay.
41. **Assistant:** [selects San Francisco]
42. **User:** What next?
43. **Assistant:** Let's remove an airline constraint.

Figure 2: Excerpt from an assistant scenario.

When a task is proposed by the user or the agent (as in lines 36 and 43 respectively), either collaborator can perform the action (the user does the action in line 38; the agent in line 41). The agent can suggest what to do next (line 43) based on its recipe library.

2. INCORPORATING TUTORIAL STRATEGIES

The most fundamental difference between tutoring and assisting is that for assistants the main shared objective is to get the task done

(a task goal), whereas for tutors the main objective is for the user to *learn* how to get the job done (a learning goal). When the student wants to learn as well, this goal is shared.

Learning goals and task goals may be interleaved and may sometimes be in service of and sometimes be in conflict with one another. For example, the goal of keeping the student's motivation level high might conflict with the overall learning goal, because it suggests helping the student when she gets stuck rather than making her perform every action herself. An assistant may sometimes form a subsidiary goal of teaching the user during task performance. The assistant may want to make sure that the user is able to do the task herself next time. In that case, the learning goal is in service of the task goal.

The differences between the objectives of tutoring and assisting required us to add functionality to the assistant in order for it to also be a tutor. Furthermore we discovered that many of the necessary differences could be implemented in COLLAGEN as adjustable parameters, whose values determine how assistive or tutorial the agent is, as well as the tutoring style. In this section we describe how these differences were encoded in Triton's parameters. Like the original air travel assistant, Triton is an agent built with COLLAGEN.

Fig. 3 is a scenario demonstrating some tutoring behaviors that we will use as an example. In this scenario, "What next?" is a generic help button the user can press when unsure of what to do. The constraints of the flight were communicated to the user earlier in the dialogue. In this scenario the user asks for help a lot; this is mostly to show Triton's functionality.

17. **User:** What next?
18. **Triton:** Please specify the route of the trip.
19. **User:** What next?
20. **Triton:** Select Atlanta and then select Boston.
21. **User:** What next?
22. **Triton:** Please select Atlanta.
23. **User:** What next?
24. **Triton:** [points to Atlanta with cursor] Please click on the dot near Atlanta on the map.
25. **User:** [clicks on Atlanta] What next?
26. **Triton:** Please select Boston.
27. **User:** [clicks on Dallas]
28. **Triton:** Oops! What you did was select Dallas; what you should have done was select Boston. I will bring you back to the previous state by clicking undo. [undo] Please select Boston.
29. **User:** [clicks on Boston]
30. **Triton:** Great job!
31. **User:** What next?
32. **Triton:** Select the dates and times of travel.

Figure 3: Excerpt from a tutorial scenario in which Triton is leading the user through a worked example of planning a flight from Atlanta to Boston.

The user could perform many correct actions in a row, and Triton would follow along without interrupting.

2.1 Worked Examples

Tutors and assistants often differ in what they know about tasks. While an assistant usually has *generic recipes*, a tutor often relies

heavily on *worked examples*. A generic recipe has many unspecified parameters such as what city to select, whereas the recipe for a worked example has specific bindings, such as selecting Atlanta. Worked examples are particularly important for artificial tutors, for which it might be difficult to automatically generate explanations and error recovery strategies from purely generic knowledge.

Our worked example recipes in Triton were developed as extensions of the generic recipes in the original air travel assistant.

2.2 The User Is Not Always Right

When there is a conflict of opinion between the agent and the user regarding what should be done next, an assistant usually trusts the user's judgment. This is not so much the case with tutoring, where the student is less likely to be right, and where correcting errors is important.

2.2.1 Determining When a Task is Completed

If the user says she is done with a task, our original assistant assumed the goal is satisfied. In a tutoring situation it is the tutor who ultimately decides when the task is correctly completed.

2.2.2 Responding to Errors

COLLAGEN interprets user actions with reference to its recipe library. Using its plan recognition capability [4], it can understand when a user begins working on a new task. If the user is doing something that is not a part of a known recipe, COLLAGEN treats it as an unrecognizable action.

When the user starts doing something the agent doesn't understand, an assistant will usually let the user proceed without interruption. But the more a user doesn't understand what she is doing, the more likely that an interruption is actually an *error*. Tutors intervene to get students back on track.

In Triton, the point at which the agent intervenes is controlled by a parameter, n , which is the number of unrecognized actions to allow. We chose as Triton's current default behavior, based on Anderson et. al. [1], to intervene as soon as a mistake is detected (i.e. $n = 0$).

New response mechanisms were also added to COLLAGEN to support error remediation. Triton first tells the user what she did and reminds her of what she should have done. Then it uses the "undo" button, to put the application back into the previous state, and reminds her of what to do next (e.g., Fig. 3, line 28).

2.3 Tutors Are Not Maximally Helpful

An assistant will usually try to be maximally helpful with the task at hand. In contrast, a tutor may choose not to be helpful because the goal of having the student learn outweighs the goal of getting the task done. This broad difference has the following consequences.

2.3.1 Waiting for Student Initiative

If an assistant sees that a particular action needs to be done next, there is no reason for it not to propose that action immediately. A tutor, on the other hand, may want to wait and give the student a chance to come up with the next action on her own, or ask for help if she is really stuck. The level of *initiative* was already a parameter of COLLAGEN. Its behavior needed to be refined somewhat, however, to better support tutorial dialogue.

2.3.2 Suggesting Actions Without Doing Them

When there is an action to be done, an assistant should be willing to do it. A tutor, however, may decide that the student must do the action herself in order to learn effectively. In Triton, a parameter determines whether either the user or the agent or just the user performs application actions.

2.3.3 Explaining

Assistants and tutors both should be able to suggest what to do next. In Triton, the user gets more help as she asks for it. At first, Triton gives a task description of what to do next (e.g., Fig. 3, line 18). This is called *proposing an action*. If the user asks for more help, Triton returns a more detailed explanation of what to do (e.g., Fig.3, lines 20 and 24).

In Triton, an *explanation* consists of any number of utterances, pointing acts or demonstrations. Explaining behavior is implemented using recipes, which are hierarchical structures of *actions*: (1) *composite actions*, which can be decomposed into other actions, and (2) *primitive actions*, which typically correspond to single clicks on the application GUI. Fig. 4 shows the part of the recipe used in Fig. 3.

It is a common strategy in ITS's to initially respond to students' help requests by giving a high-level hint, and then to provide more specific information if the student asks for more help [1, 2]. Triton uses this strategy. The first request for help results in a task description. The second request results in an explanation. Each action has an explanation associated with it stored in an *explanation recipe*. The explanation recipe for a composite action is an automatically generated utterance listing the task descriptions of the actions composing it. The explanation recipe for a primitive action is an application-level description of what to do on screen (e.g. "Click on the dot near Atlanta on the map").

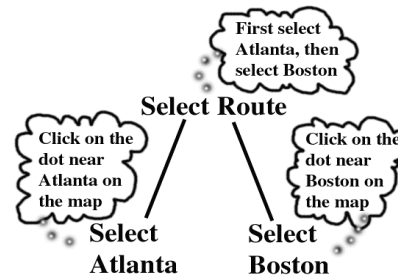


Figure 4: The recipe for selecting the route in a worked example. The thought balloons are the explanation recipes associated with the actions.

As the user asks for more help, Triton goes on to the next action. If the previous action was composite, then the next action will be its first component. The idea is that if the student does not understand how to do the composite action even after hearing what it is composed of, what she needs to hear next is how to do the first component. In Fig. 3, line 21, the user asks for more help after the breakdown explanation has been given. Triton responds by moving down the action composition hierarchy to the next action, which in this case is a primitive. The task description is given first in Fig. 3, line 22. The user asks for help once again in line 23, and then the explanation for the primitive action, an application-level instruction, is given in line 24.

2.3.4 Demonstrating

Sometimes a tutor might want to *demonstrate* how something is done. For Triton this means performing a sequence of actions, followed by a sequence of undo actions to put the application in its previous state. We put demonstrations in the explanation recipes: Triton does any actions that are in explanation recipes (including, usually, the appropriate number of undo's). This simple change generates the desired demonstration behavior. Both task descriptions and explanation recipes were a part of the existing COLLAGEN architecture.

2.3.5 Pointing

The original assistant would always use its cursor (the white hand in the upper right of Fig. 1) to point to the part of the application to be worked on when proposing actions. The assistant was parameterized so that when tutoring, it would point at the correct place in the application at a better time: when explaining a primitive action.

3. CONCLUSION

Intelligent tutors and assistants have much in common. There has been a great deal of research in both tutoring [8, 9] and assisting. The goal of this work is to bridge these two areas. Triton provides both modes of interaction, showing that a single architecture based on collaborative discourse theory can support them both. In natural settings, both tutoring and assisting are mediated by dialogue [3, 6], shared goals, and plans. Future work can show how tutors can smoothly transform into assistants, and assistants can tutor when needed.

This work makes two main contributions to the ITS field. The first is a middleware architecture that is independent of both the domain being taught and the pedagogical style used. The second is the use of recipes as a single representational structure that can accommodate abstract actions, explanations, and worked examples, including GUI actions, pointing actions, utterances, explanations, and demonstrations. A new error response system had to be added to COLLAGEN. For tutoring strategies to take place, some recipes must be written for worked examples and explanations.

All of the other behavioral differences between Triton and the original assistant were made into parameters: (1) When to

intervene after error detection (after how many unrecognized steps), (2) who should default to do the task actions (the user, or anyone), and (3) when to point (upon proposal or explanation).

In the future, among many other open questions, we would like to investigate how to use a student model [8] to inform parameter value changes so it can shift smoothly from a tutor to an assistant role.

4. REFERENCES

- [1] Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2), 167—207.
- [2] Gertner, A.S. Conati, C, and VanLehn, K. (1998). Procedural help in Andes: Generating hints using a bayesian network student model. In *Proc. of the 15th National Conf. Artificial Intelligence*.
- [3] Graesser, A. C., Person, N. K., and Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychology*, 9, 495—522.
- [4] Lesh, N, Rich, C. and Sidner, C. L. (1998). Using plan recognition in human-computer collaboration, *Seventh Int. Conf. on User Modeling*, Banff, Canada.
- [5] Lochbaum, K. E. (1998). A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4). 525—72.
- [6] Moore, J. D. and Paris, C. L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information, *Computational Linguistics*, 19(4), 651—695.
- [7] Rich, C. and Sidner, C. L. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, Vol. 8, No. 3/4, 315—350.
- [8] Sleeman, D. H., and Brown, J. S. (Eds.) (1982). *Intelligent Tutoring Systems*. Academic Press, London.
- [9] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, Inc. Los Altos, CA.